REMARKS

Applicant has amended claims 1 and 4 – 6. Applicant has also added new claim 7 – 15. Applicant respectfully requests reconsideration of this application in view of the following remarks.

35 USC §112 Rejections

The Office Action rejects claims 1 – 6 per 35 USC §112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Specifically, the Office Action states that claim 1 includes unclear language in steps (C) and (D). Step (D) of claim 1 has been amended to satisfy 35 USC §112 without narrowing the scope of claim 1. The Office Action also states that language in claims 4 and 5 is unclear. Amendments that do not narrow the scope of claims 4 – 6 have been made to satisfy the requirements of 35 USC §112.

In claim 1, the Office Action states that the term 'set' is unclear. Applicant respectfully submits that the term 'set' is clear as it appears in claim 1. Step (C) of claim 1 states, "determining the operations required to set each component of the state at each selected program point" (emphasis added). An operation may set a component in a number of ways. To illustrate by example, an operation sets a component by setting the component to a given value, setting a field of the component to a value, setting all fields of a component to a given value, destructing the component, creating the component, etc. These examples to illustrate setting a component are only by example and not meant to be limiting upon the claimed invention. Using a particular one of these examples would unnecessarily limit the scope of claim 1 to less than Applicant's invention.

35 USC §103 Rejections

The Office Action rejects claims 1 under 35 USC §103 as being unpatentable over "How Debuggers Work", Jonathon B. Rosenberg, 1996 (the Rosenberg reference) in view of U.S. Patent 6,446,258 to McKinsey et al. (the McKinsey reference). Applicant respectfully submits that the Rosenberg reference in view of the McKinsey reference does not teach Applicant's

claimed invention and that the Office Action mistakenly equates tracing a stack for debugging with Applicant's claimed invention.

The Office Action mistakenly argues that steps (A) – (C) of claim 1 are described in the Rosenberg reference. Independent claims 1, 7 and 10 include the following elements:

(A) analyzing a program to determine the state of said data structure at selected program points;

(B) partitioning into components;

(C) determining the operations to set the components.

The Rosenberg reference defines a stack trace as "a list of the procedure activation records or frames currently on the stack" (p. 136). The Rosenberg reference describes unwinding "the stack to find the parent procedure's frame pointer and return address" (p. 136) and **not** to determine the state of a data structure as in step (A) of claims 1, 7, and 10. The Rosenberg reference does not discuss partitioning or components as found in step (B) of claims 1, 7, and 10. In addition, the Rosenberg reference never discusses step (C) as found in claims 1, 7, and 10. Instead, the Rosenberg reference describes each procedure call pushing return addresses on a stack and child procedures pushing their parent's frame pointer address onto the stack before altering the frame pointer and the stack pointer (p.137), **which is not** "determining the operations required to set each component of the state at each selected program point" as in claim 1 or "computing placement of the set of operations to eliminate partial redundancies" as in claims 7 and 10.

Applicant respectfully asserts that the Rosenberg reference does not teach at least elements (A) – (C) of the independent claims 1, 7, and 10 and that the combination of the Rosenberg reference and the McKinsey reference does not teach Applicant's claimed invention.

Applicant respectfully traverses all rejections made with respect to the dependent claims. Applicant asserts that all dependent claims of the application are dependent on one of the above allowable independent claims.

CONCLUSION

Applicant respectfully requests reconsideration of this application as amended. Applicant respectfully submits that the Claims are novel, and nonobvious over the cited prior art for the above reasons and are in condition for allowance. Accordingly, Applicant respectfully requests the rejections be withdrawn and the Claims be allowed.

INVITATION FOR A TELEPHONE INTERVIEW

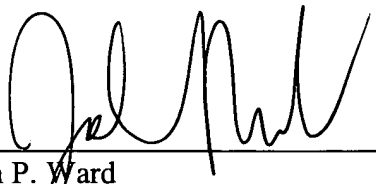
If the Examiner believes a telephone conference would expedite or assist in the allowance of the present application, the Examiner is invited to call John Ward at (408) 720-8300, x237.

CHARGE OUR DEPOSIT ACCOUNT

Please charge any shortage in connection with this communication to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN



Date: 4/15/2003

John P. Ward
Reg. No. 40,216

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1026
(408) 720-8300 x237

VERSION WITH MARKINGS TO SHOW CHANGES MADE

IN THE CLAIMS

New claims 7 – 15 have been added.
Claims 1 and 4 – 6 have been amended as follows:

1. (Amended) For a computer-executable program that operates on a data structure, where the data structure must have a required state at selected program points, a method of transforming said program comprising the steps of:

(A) analyzing the program to determine the state of said data structure at said selected program points;

(B) partitioning said determined state at each said program point into components that may each be set separately;

(C) determining the operations required to set each component of the state at each selected program point; and

(D) placing said operations ~~in a way that~~ to eliminate~~s~~ partial redundancies of said operations.

2. (Unamended) The method of claim 1, wherein the data structure stores items on a first-in-last-out basis.

3. (Unamended) The method of claim 2, wherein the states of the data structure are represented as paths on a tree of nodes where:

(A) each path traverses the tree towards the root; and

(B) each node on the path represent a component of the state.

4. (Amended) The method of claim 2, wherein the data structure represents actions to be taken by the program if an exception~~al~~ occurs ~~situation arises~~.

5. (Amended) The method of claim 4, wherein the selected program points are the points of execution immediately before instructions that might cause an exceptional ~~situation~~.

6. (Amended) The method of claim ~~4~~5, further comprising representing ~~wherein~~ the actions to be taken ~~are represented explicitly~~ as exceptional paths in a graph ~~before the transformation, and said exceptional paths are removed~~.

7. (New) For a computer-executable program that operates on a data structure, where the data structure must have a required state at selected program points, a method of transforming said program comprising the steps of:

(A) analyzing the program to determine the state of an instance of said data structure at said selected program points;

(B) partitioning said instance of said data structure into components;

(C) determining a set of one or more operations for setting the components;

(D) computing placement of the set of operations to eliminate partial redundancies; and

(E) inserting the set of operations at said program points according to the computed placement.

8. (New) The method of claim 7 wherein the data structure is an exception handling stack.

9. (New) The method of claim 8 wherein the components are a pointer to the exception handling stack and an exception handling data structure.

10. (New) A machine-readable medium having a set of instructions, which when executed by a set of one or more processors, causes said set of processors to perform operations comprising:

(A) analyzing a program that operates on a data structure, which must have a required state at selected program points in the program, to determine the state of an instance of said data structure at said selected program points;

(B) partitioning said instance of said data structure into components;

(C) determining a set of one or more operations for setting the components;

(D) computing placement of the set of operations to eliminate partial redundancies; and

(E) inserting the set of operations at said program points according to the computed placement.

11. (New) The machine-readable medium of claim 10, wherein the data structure stores items on a first-in-last-out basis.

12. (New) The machine-readable medium of claim 11, wherein the states of the data structure are represented as paths on a tree of nodes where:

(A) each path traverses the tree towards the root; and

(B) each node on the path represent a component of the state.

13. (New) The machine-readable medium of claim 11, wherein the data structure represents actions to be taken by the program if an exception occurs.

14. (New) The machine-readable medium of claim 13, wherein the selected program points are the points of execution immediately before instructions that might cause an exception.

15. (New) The machine-readable medium of claim 14, wherein the actions to be taken are represented explicitly as exceptional paths in a graph before the transformation, and said exceptional paths are removed.